Библиотека пользовательского интерфейса модуля LTR34

Крейтовая система LTR

Руководство программиста

Ревизия 1.0.2

Июнь 2009г.

Автор руководства:
Милованов А.Н.

ЗАО "Л-КАРД"
117105, г. Москва, Варшавское ш., д. 5, корп. 4, стр. 2

тел.: (095) 785-95-25
факс: (095) 785-95-14

Адреса в Интернет:
http://www.lcard.ru/
ftp://ftp.lcard.ru/pub

E-Mail:
Отдел продаж: sale@lcard.ru
Техническая поддержка: support@lcard.ru
Отдел кадров: job@lcard.ru
Общие вопросы: lcard@lcard.ru

Представители в регионах:
Украина: HOLIT Data Sistems, http://www.holit.com.ua, (044) 241-6754
Санкт-Петербург: Autex Spb Ltd., http://www.autex.spb.ru, (812) 567-7202
Новосибирск: Сектор-Т, http://www.sector-t.ru, (383-2) 396-592
Екатеринбург: Аск, http://www.ask.ru, 71-4444
Казань: ООО 'Шатл', shuttle@kai.ru, (8432) 38-1600

Крейтовая система LTR

История ревизий настоящего документа.

Ревизия	Дата	Примечания по внесенным изменениям
1.0.0	26.07.2006	Первая доступная для пользователя ревизия
1.0.1	23.03.2007	Руководство переработано в соответствии со значительными изменениями в API модуля
1.0.2	10.06.2009	Исправлены небольшие несоответствия между API и руководством

На CD-ROM, входящий в комплект поставки, всегда записывается последняя ревизия данного документа. Кроме того, последнюю ревизию Вы сможете найти в разделе *библиотека файлов* на нашем сайте.

L-Card оставляет за собой право обновлять документацию без уведомления пользователей об изменениях.

Содержание :

1 Основные принципы работы с библиотекой пользовательского интерфейса модуля LTR34

Модули LTR34-4 и LTR34-8 являются соответственно 4- и 8-канальными цифроаналоговыми преобразователями (ЦАП) и предназначены для широкого круга задач, где требуется качественное воспроизведение постоянного или переменного напряжения. Генерация сигнала может производиться в синхронном или асинхронном режимах, с потоковым выводом данных (до 500 килосэмплов в секунду), либо в режиме автогенератора периодического сигнала с предварительной накачкой в буфер ЦАП цифрового сигнала размером до 2 млн сэмплов. Область применения модуля – задачи, где требуется качественное воспроизведение сигнала в синхронном или асинхронном режиме. Подробно о структуре модуля и всех его возможностях написано в документе «Крейтовая система LTR. Руководство пользователя». В данном Руководстве речь пойдет о программировании модуля посредством вызова функций, содержащихся в библиотеке пользовательского интерфейса.

На плате модуля отсутствует микроконтроллер, модуль полностью управляется ПЛИС. В связи с этим необходимо как можно точнее соблюдать правила работы с ним. На плате модуля также присутствует флэш-память на 2 Кб, используемая для хранения калибровочных данных и идентификационной информации модуля.

Запуск генерации сигналов возможен как от внутреннего, так и от внешнего старта.

Для поддержания непрерывной генерации на плате имеется FIFO буфер объемом 8 Мбайт.

Последовательность применения пользовательских функций сходна с использованием аналогичных функций в программном обеспечении других модулей системы LTR, хотя имеет свои особенности в связи с направлением основного потока данных от крейта к модулю. В общих чертах эта последовательность выглядит следующим образом:

- Инициализация интерфейсного канала связи с модулем, установка настроек по умолчанию и открытие модуля;
- Установка параметров работы (параметры частоты ЦАП, загрузка данных в буфер модуля), передача этих параметров модулю;
- Первичная загрузка данных в FIFO;
- Старт ЦАП;
- Если не используется режим автогенерации, то необходима подкачка данных в ЦАП;
- Останов работы ЦАП;
- Закрытие интерфейсного канала связи с модулем.

Функции библиотеки модуля LTR34 разрабатывались с использованием основных функций крейта LTR.

2 Описание работы с пользовательской библиотекой API-функций

Функции, входящие в состав пользовательской библиотеки модуля LTR34, можно разделить на следующие группы:

Функции инициализации и открытия – это функции, предназначенные для инициализации и открытия интерфейса с модулем:
LTR34_Init();
LTR34_Open();
LTR34_Reset();
LTR34_IsOpened().

Функции конфигурирования – предназначены для настройки модуля.
LTR34_CreateLChannel();
LTR34_SetConfig().

Функции запуска и остановки генерации сигнала
LTR34_DACStart();
LTR34_DACStop().

Для считывания текущей фабричной калибровки модуля также присутствуют функции
LTR34_GetCalibrCoeffs()

Функции подготовки и передачи данных предназначены для заполнения FIFO модуля и отправки команд в модуль

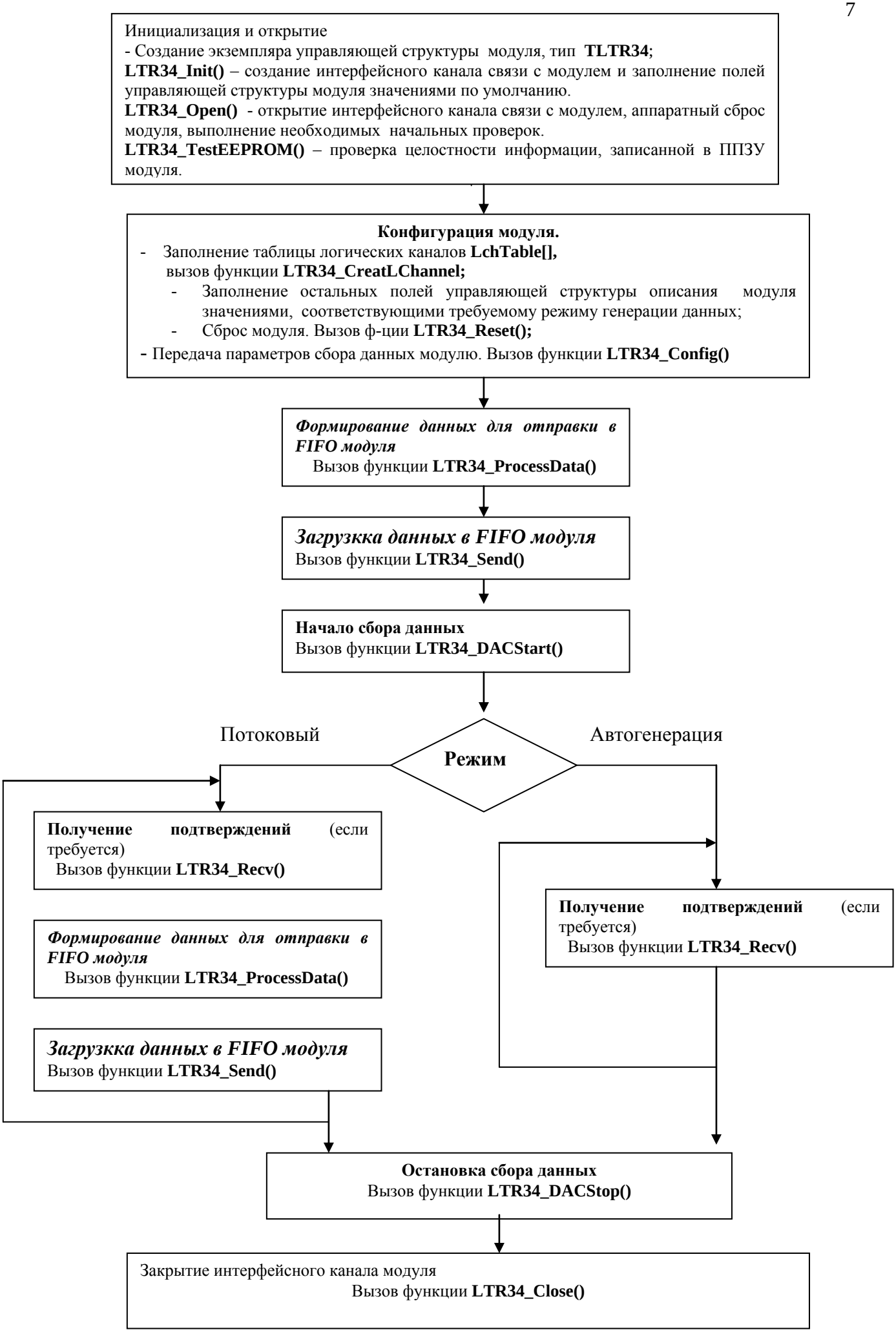
LTR34_Send();
LTR34_ProcessData().

Функция приема команд и данных:
LTR34_Recv()

Дополнительные функции :
LTR34_TestEEPROM();
LTR34_GetErrorString() – получение описания ошибки по ее коду.

Функция закрытия:
LTR34_Close()

Ниже представлена типовая последовательность вызова интерфейсных функций при работе с модулем:


Инициализация и открытие
- Создание экземпляра управляющей структуры модуля, тип TLTR34;
LTR34_Init() – создание интерфейсного канала связи с модулем и заполнение полей управляющей структуры модуля значениями по умолчанию.
LTR34_Open() - открытие интерфейсного канала связи с модулем, аппаратный сброс модуля, выполнение необходимых начальных проверок.
LTR34_TestEEPROM() – проверка целостности информации, записанной в ППЗУ модуля.
Конфигурация модуля.
- Заполнение таблицы логических каналов LchTable[], вызов функции LTR34_CreatLChannel;
- Заполнение остальных полей управляющей структуры описания модуля значениями, соответствующими требуемому режиму генерации данных;
- Сброс модуля. Вызов ф-ции LTR34_Reset();
- Передача параметров сбора данных модулю. Вызов функции LTR34_Config()
Формирование данных для отправки в FIFO модуля
Вызов функции LTR34_ProcessData()
Загрузкка данных в FIFO модуля
Вызов функции LTR34_Send()
Начало сбора данных
Вызов функции LTR34_DACStart()
Потоковый
Режим
Автогенерация
Получение подтверждений (если требуется)
Вызов функции LTR34_Recv()
Формирование данных для отправки в FIFO модуля
Вызов функции LTR34_ProcessData()
Загрузкка данных в FIFO модуля
Вызов функции LTR34_Send()
Получение подтверждений (если требуется)
Вызов функции LTR34_Recv()
Остановка сбора данных
Вызов функции LTR34_DACStop()
Закрытие интерфейсного канала модуля
Вызов функции LTR34_Close()

3 Подробное описание библиотеки ltr34api

Для получения возможности вызова интерфейсных функций библиотеки ***ltr34api.dll*** из Вашего приложения необходимо выполнить следующее:

- создать проект в какой-либо из сред разработки;

- поместить в папку проекта или в папку, описанную в переменной окружения PATH, файл ltr34api.dll.

- добавить в проект информацию о способе вызова интерфейсных функций dll-библиотеки и используемых типах данных. В различных средах разработки последовательность действии и приложенные усилия могут несколько отличаться:

-

Borland C++/Borland C++ Builder :
подключить к проекту файлы LTR\LIB\BORLAND\ltr34api.lib и LTR\INCLUDE\ltr34api.h;

Microsoft Visual C++ :
подключить к проекту файлы LTR\LIB\MSVC\ltr34api.lib и LTR\INCLUDE\ltr34api.h;

Другие среды разработки :
следует обратиться к соответствующей документации на средство разработки.

Затем следует создать и добавить в проект файл с исходным текстом будущей программы; после этого можно писать свою программу, вызывая соответствующие интерфейсные функции dll-библиотеки.

3.1 Управляющая структура модуля

Поля данной структуры содержат информацию о конфигурации АЦП модуля, режимах работы модуля, диапазонах каналов, описание модуля, и его калибровки, как фабричные, так и пользовательские. Первоначальная конфигурация структуры заполняется с помощью функции LTR34_Init(). Определение структуры приводится ниже:

```
typedef struct
{
  int  size;                  // Размер структуры
  TLTR  Channel;              // Интерфейсный канал связи с модулем
  DWORD  LChTbl[8];           // Таблица лоигческих каналов
  BYTE  FrequencyDivisor;     // делитель частоты дискретизации 0..60 (31.25..500 кГц)
  BYTE  ChannelQnt;           // Определяет число активных каналов (1, 2, 4, 8)
  bool  UseClb;               // флаг применения калибровочных коэффициентов
  bool  AcknowledgeType;      // тип подтверждения false - высылать подтверждение каждого
                              // слова, true - высылать состояние буфера каждые 1024 слова
  bool  ExternalStart;        // внешний старт true - внешний старт, false - внутренний
  bool  RingMode;             // режим кольца  true - режим кольца, false - потоковый режим
  bool  BufferFull;           // статус - буфер переполнен - ошибка
  bool  BufferEmpty;          // статус - буфер пуст - ошибка
  bool  DACRunning;           // статус - запущена ли генерация
  float  FrequencyDAC;        // частота, на которую настроен ЦАП , Гц
```

```
    TINFO_LTR34  ModuleInfo;// идентификационная информация модуля
    DAC_CHANNEL_CALIBRATION  DacCalibration;  //калибровочные коэффициенты

} TLTR34;
```

Далее приводится описание полей этой структуры.

Таблица 1

Поле	Тип	Описание
size	**INT**	объем памяти, выделяемый под структуру
Channel	**TLTR**	структура, представляющая собой описание интерфейсного канала связи с крейт-контроллером;
LChTbl[]	**DWORD**	Таблица логических каналов. Определяет, какие каналы и выходы ЦАП используются в данном приложении
FrequencyDiviser	**byte**	Определяет частоту дискретизации ЦАП. Подробнее об этом см. в гл. 4.3.
ChannelQnt	**byte**	Определяет количество каналов ЦАП, задействованных в данном приложении. Может принимать значения только 1,2,4,8.
UseClb	**bool**	Определяет, применяются ли фабричные калибровочные коэффициенты при генерации данных. **false** - коэффициенты не применяются, **true** - коэффициенты применяются.
AcknowledgeType	**INT**	Тип подтверждений, высылаемых модулем при генерации сигнала. **false** – высылает подтверждение после вывода на ЦАП каждого сэмпла из выходного FIFO модуля **true** – каждые 1024 сэмпла высылает информацию о состоянии FIFO-буфера, причем это подтверждение получает только сервер, а до модуля оно не доходит
ExternalStart	**bool**	Определяет, используется внутренний или внешний старт. **false** – используется внутренний старт; **true** – используется внешний старт.
RingMode	**bool**	Определяет, используется режим автогенерации («кольца») или потоковый режим вывода на ЦАП. В режиме автогенерации достаточно однократно заполнить выходной FIFO-буфер модуля требуемыми значениями. При старте модуля значения будут выводиться на ЦАП в кольцевом режиме, т.е. дойдя до конца буфера, устройство переходит к его началу и вновь последовательно выводит все сэмплы, находящиеся в буфере. При потоковом режиме необходимо постоянно подкачивать данные в FIFO модуля по мере их вывода на ЦАП. Подробнее об этом см. в гл. 6.2. **true** – режим автогенерации **false** – потоковый режим
BufferFull	**bool**	Флаг переполнения FIFO-буфера модуля. **false** – буфер не переполнен **true** - буфер переполнен

BufferEmpty	**bool**	Флаг, указывающий, что выходной буфер пуст. **false** – буфер не пуст **true** - буфер пуст
DACRunning	**bool**	Флаг, указывающий, что генерация сигнала ЦАПом запущена. **false** – Генерация остановлена **true** - Генерация запущена
FrequencyDAC	**float**	Частота дискретизации, на которую настроен ЦАП в текущей конфигурации. Следует отметить, что это общая частота дискретизации ЦАП, не зависит от количества активных каналов. Поле обновляется после вызова функции **LTR34_Config()**.
ModuleInfo		Идентификационная информация модуля. Тип – **TINFO_LTR34**. Данный тип представляет собой структуру. Ее описание приводится ниже в этой главе настоящего Руководства.
DacCalibration		Структура, содержащая массивы, в которые после открытия модуля загружаются заводские калибровочные коэффициенты. Тип - **DAC_CHANNEL_CALIBRATION**. Данный тип представляет собой структуру. Ее описание приводится ниже в этой главе настоящего Руководства.

В поле подструктуры **ModuleInfo** типа **TINFO_LTR34** после открытия модуля считываются из ППЗУ идентификационные данные модуля. Описание типа **TINFO_LTR34** приводится ниже:

```
typedef struct
{

  CHAR Name[16];              // имя модуля
  CHAR Serial[24];            // серийный номер модуля
  CHAR FPGA_Version[8];       // Версия прошивки ПЛИС
  CHAR CalibrVersion[8];      // Версия калибровки
   BYTE MaxChannelQnt;        // Максимальное кол-во каналов (4 или 8)

} TINFO_LTR34,*PTINFO_LTR34;
```

Подструктура для хранения заводских калибровочных коэффициентов

В поле подструктуры **DacCalibration** типа **DAC_CHANNEL_CALIBRATION** после открытия модуля считываются из ППЗУ считываются из ППЗУ калибровочные коэффициенты модуля. Описание типа **DAC_CHANNEL_CALIBRATION** приводится ниже:

```
typedef struct
{

   float FactoryCalibrOffset[2*LTR34_DAC_NUMBER_MAX];  // к-ты смещения
   float FactoryCalibrScale[2*LTR34_DAC_NUMBER_MAX];   // к-ты усиления

 } DAC_CHANNEL_CALIBRATION.
```

3.2 Описание функций библиотеки ltr34api.dll

Функции библиотеки ltr34api.dll используются совместно со штатными функциями ltrapi.dll.

Для удобства использования библиотеки введены локальные типы данных :

```
typedef BYTE byte;

#ifndef __cplusplus
        #define true 1
        #define false 0
        typedef byte bool;
#endif
```

Формат: INT LTR34_Init(TLTR34 *module)
Параметр: **module** - указатель на структуру типа **TLTR34**, которая должна быть объявлена перед вызовом функции

Описание: Выполняет инициализацию интерфейсного канала связи с модулем и заполнение полей управляющей структуры модуля значениями по умолчанию. Ниже приводятся значения, которыми данная функция инициирует поля указанной структуры:
 	
module->NumberOfChannel=1; // Используется один канал
module->AcknowledgeType=true; // Высылать состояние буфера каждые 100 мс
module->ExternalStart=false; // Внешний старт не используется, применен внутренний
module->RingMode=false; // Режим кольца отключен, используется потоковый вывод
module->BufferFull=false; // Флаг «буфер переполнен» обнуляется
module->BufferEmpty=false; // Флаг «буфер пуст» обнуляется
module->DACRunning=false; // Флаг «Генерация запущена» обнуляется
module->FrequencyDAC=500000f; // Частота дискретизации ЦАП – 500 кГц
 	
LChTbl[0]=LTR34_CreateLChannel(1,0); // Создаем один логический канал,
// соответствующий. физическому каналу 1, выход 1:1
 	
/* До вызова ф-и LTR34_Open() следующие поля – пустые строки */
 	
strcpy((CHAR *)module->ModuleInfo.Serial, "\0");
strcpy((CHAR *) module ->ModuleInfo.FPGA_Version, "\0");
strcpy((CHAR *) module ->ModuleInfo.CalibrVersion, "\0");
strcpy((CHAR *) module ->ModuleInfo.Name, "\0");
ModuleInfo.MaxChannel=4;

Возвращаемое значение: код ошибки, тип INT. Если “0” – функция выполнилась без ошибок

Формат: INT LTR34_Open(TLTR34 *module, DWORD net_addr, WORD net_port, CHAR *crate_sn, INT slot_num, CHAR *biosname)
Параметры: - **module** – указатель на структуру описания модуля (тип TLTR34) - **net_addr** – сетевой адрес сервера A.B.C.D в формате HEX: 0xABCD. Например, **net_addr** для адреса 127.0.0.1 будет выглядеть следующим образом: 0x7F000001. Необходимо помнить, что все компоненты адреса должны иметь значение, не превосходящее 255. - **net_port** – сетевой порт сервера - **crate_sn** – серийный номер *крейта*. - **slot_num** – номер посадочного места в крейте, где расположен модуль (нумерация *с единицы*!)
Описание: Функция открывает интерфейсный канал связи с модулем, считывает из ППЗУ модуля его идентификационную запись и калибровочные коэффициенты. После работы функции в соответствующих полях подструктуры **ModuleInfo** будут находиться: имя модуля, серийный номер модуля, версия прошивки ПЛИС, версия калибровки и значение максимального количества каналов (4 или 8). В соответствующих полях подструктуры **DacCalibration** будут находиться фабричные калибровочные коэффициенты всех каналов модуля.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок. Если возвращаемое значение равно –10, то это предупреждение, что интерфейсный канал уже открыт. Тем не менее функция выполнилась успешно и можно продолжать работу. Однако дальнейшая работа модуля может быть некорректной. Рекомендуется разобраться в причинах предупреждения и закрыть открытый ранее канал.

Формат: INT LTR34_Close(LTR34 *module)
Параметр: **module** – указатель на структуру описания модуля, тип TLTR34
Описание: Выполняет закрытие интерфейсного канала связи с модулем. Эту функцию следует вызывать *всегда* перед окончанием работы с модулем.
Возвращаемое значение: код ошибки, тип INT. Если “0” – функция выполнилась без ошибок

Формат: INT LTR34_IsOpened(TLTR34 *module)
Параметр: **module** – указатель на управляющую структуру модуля, тип TLTR34
Описание: Функция позволяет отслеживать состояние соединения с модулем: если возвращаемый результат отличен от 0, то соединения нет.
Возвращаемое значение: Если “0” – интерфейсный канал связи с модулем создан и открыт. Если значение ненулевое - канал не создан

Формат: INT LTR34_Reset(TLTR34 *module)
Параметр: **module** – указатель на управляющую структуру модуля, тип TLTR34
Описание: Функция осуществляет аппаратный сброс модуля и очистку FIFO-буфера ЦАП. Эту функцию следует вызывать ***всегда*** перед каждым новым сеансом конфигурации модуля, загрузки и генерации данных. Если этого не сделать, то старт сбора данных может быть неудачным. Особенно при потоковой генерации. Это обусловлено аппаратными особенностями модуля.
Возвращаемое значение: код ошибки, тип INT. Если “0” – функция выполнилась без ошибок

Формат: INT LTR34_Config(TLTR34 *module)
Параметр: **module** – указатель на управляющую структуру модуля, тип TLTR34
Описание: Функция осуществляет передачу модулю значений параметров из полей управляющей структуры. Перед вызовом этой функции все поля управляющей структуры должны быть заполнены правильными значениями исходя из требуемого режима генерации данных. Во время работы этой функции модулю передаются следующие параметры: количество активных каналов, тип подтверждений, тип старта (внешний или внутренний), а также тип генерации данных: автогенерация (кольцевой) или потоковый (непрерывная передача данных модулю в процессе генерации). После работы данной функции поле **FrequencyDAC** управляющей структуры принимает значение фактической частоты дискретизации ЦАП. Частота генерации *для каждого канала* будет равна **FrequencyDAC**, поделенной на количество задействованных каналов. Т.е. **FrequencyDAC/ChannelQnt.** *При выполнении функции **LTR34_Config()** также происходит очистка FIFO-буфера.*
Возвращаемое значение: код ошибки, тип INT. Если “0” – функция выполнилась без ошибок

Формат: INT LTR34_DACStart(TLTR34 *module)
Параметр: **module** – указатель на управляющую структуру модуля, тип TLTR34
Описание: Запуск генерации сигнала ЦАП. Перед первым стартом генерации вне зависимости от режима требуется первоначальная загрузка FIFO-буфера. При остановке модуля функцией **LTR34_DACStop()** FIFO-буфер модуля не очищается, и генерацию можно повторить, не загружая данные снова. Особенно это касается режима автогенерации. Если же была произведена конфигурация модуля (функция **LTR34_Config()**) или аппаратный сброс модуля (функция **LTR34_Reset**), то FIFO буфер очищается, и последующая генерация должна обязательно предваряться загрузкой данный в FIFO-буфер.
Возвращаемое значение: код ошибки, тип INT. Если “0” – функция выполнилась без ошибок

Формат: INT LTR34_DACStop(TLTR34 *module)
Параметр: **module** – указатель на управляющую структуру модуля, тип TLTR34
Описание: Останов генерации сигнала ЦАП. FIFO-буфер модуля при этом *не очищается*, и если установлен режим автогенерации, то можно снова запускать генерацию без дополнительной подкачки FIFO буфера.
Возвращаемое значение: код ошибки, тип INT. Если “0” – функция выполнилась без ошибок

Формат: INT LTR34_GetCalibrCoeffs(TLTR34 *module)
Параметр: **module** – указатель на управляющую структуру модуля, тип TLTR34
Описание: Выполняет чтение фабричных калибровочных коэффициентов из ППЗУ модуля. Считанные коэффициенты помещаются в поле **DacCalibration** управляющей структуры модуля. Данная функция также входит в состав функции **LTR34_Open().**
Возвращаемое значение: код ошибки, тип INT. Если “0” – функция выполнилась без ошибок

Формат: INT LTR34_Send(TLTR34 *module, DWORD *data, DWORD size, DWORD timeout)
Параметры: - **module** – указатель на управляющую структуру модуля, тип **TLTR34;** - ***data** – указатель на массив команд или данных, отсылаемых в модуль; - **size** – размер массива команд или данных, отсылаемых в модуль; - **timeout** – максимальное время в миллисекундах, в течение которого модуль может выполнять отправку команд или данных.
Описание: Выполняет отправку в модуль массива команд или данных, адресуемого указателем **data.** Если отправляются данные, предназначенные для генерации сигнала, то эта функция, как правило, должна применяться *после использования функции* **LTR34_ProcessData()**. Однако в некоторых случаях пользователь может сам формировать массив данных для отправки в модуль. Это несколько расширяет возможности ЦАПа, однако усложняет процесс формирование массива. Если выполняется отправка данных в FIFO-буфер, то при использовании режима автогенерации данную функцию следует выполнить только один раз перед стартом ЦАПа. Если используется потоковый режим, то данные следует периодически подгружать во время генерации и, следовательно, применять **LTR34_Send()** неоднократно. Подробнее о формировании данных и заполнении FIFO-буфера модуля см. в гл.5.
Возвращаемое значение: тип INT. Если возвращаемое значение <0, то оно представляет собой код ошибки, а функция выполнилась неудачно. Если >=0 – возвращаемое значение равно количеству отправленных слов команд или данных.

Формат: INT LTR34_ProcessData(TLTR34 *module, double *src, DWORD *dest, DWORD size, bool volt)
Параметры: - **module** – указатель на управляющую структуру модуля, тип TLTR34; - ***src** – указатель на массив с кодами, определяющими уровень генерируемого сигнала; - ***dest** – массив со словами-данными, готовыми к отправки в FIFO-буфер модуля. В этом массиве содержатся данные, формат которых приведен в соответствие с требуемым форматом работы модуля. - **size** – размер массива данных, отсылаемых в FIFO модуля; - **timeout** – максимальное время в миллисекундах, в течение которого модуль может выполнять отправку данных; - **volt** – указывает, в каких единицах задан сигнал в массиве **src**: в кодах или в вольтах: **false** – в кодах, **true** – в вольтах.
Описание: Выполняет подготовку данных к отправке в FIFO-буфер модуля. Из массива **src**, содержащего информацию непосредственно об уровне генерируемого сигнала, функция формирует массив **dest,** где данные приведены в формат, требуемый для отправки данных в модуль. Кроме кодов сигнала, к каждому сэмплу функция добавляет служебные поля: номер физического канала ЦАП, номер посадочного места модуля в крейте и др. Уровень сигнала в массиве **src** может определяться как непосредственно кодами, так и задаваться в вольтах. Если сигнал задается в вольтах, то параметр **volt** необходимо установить в **true**. Тогда функция автоматически пересчитает вольты в коды, и в выходном массиве **dest** будут содержаться уже коды сигнала. После отправки массива **dest** в FIFO-буфер модуля (посредством вызова функции **LTR34_Send()**), эти данные будут использоваться для генерации сигнала после старта ЦАП. Данные в массиве должны быть расположены по кадрам. Если сигнал задается в вольтах,

то необходимо учитывать, какой выход данного канала используется: прямой (1:1) или масштабирующий (1:10). Другими словами, выходной массив формируется из входного с учетом Таблицы Логических Каналов. Подробнее о формировании данных для загрузки в ЦАП см. в гл.5.1.
Возвращаемое значение: код ошибки, тип INT. Если “0” – функция выполнилась без ошибок

Формат: INT LTR34_Recv(TLTR34 *module, DWORD *data, DWORD *tmark, DWORD size, DWORD timeout)
Параметры: - **module** – указатель на управляющую структуру модуля, тип TLTR34; - ***data** – указатель на массив команд или данных, принимаемых из модуля; - ***tmark** – массив меток; - **size** – размер массива команд или данных, принимаемых из модуля; - **timeout** – максимальное время в миллисекундах, в течение которого модуль может выполнять прием команд или данных.
Описание: Выполняет получение массива слов из модуля размером **size** . Полученные слова при выходе из функции содержатся в массиве, адресуемом указателем **data**. Указатель **tmark** адресует массив, содержащий синхрометки (секундные и СТАРТ). **Если элементы массива tmark не используются в программе, то в качестве значения этого параметра можно использовать NULL**. Параметр **timeout** определяет максимальное время в миллисекундах, в течение которого функция будет ожидать получения заказанного количества слов. Если требуемое количество получено до истечения таймаута, функция завершается немедленно. Если по истечении таймаута не было получено требуемое количество слов, то функция все равно завершатся. Возвращаемое значение функции равно количеству полученных слов от модуля. Если же возвращаемое значение отрицательно, то это свидетельствует об ошибочном завершении. В этом случае следует идентифицировать ошибку функцией **LTR51_GetErrorString().** Если пришли данные о переполнении или опустошении FIFO-буфера ЦАП, то поля **BufferFull** и **BufferEmpty** управляющей структуры модуля обновляются. *Примечание: Описание этой функции соответствует описанию функции **LTR_Recv()** библиотеки крейт-контроллера ltrap.dll.*
Возвращаемое значение: тип INT. Если возвращаемое значение <0, то оно представляет собой код ошибки, а функция выполнилась неудачно. Если >=0 – возвращаемое значение равно количеству принятых слов команд или данных.

Формат: INT LTR34_GetCalibrCoeffs(TLTR34 *module)
Параметры: - **module** – указатель на управляющую структуру модуля, тип TLTR34;
Описание: Выполняет чтение заводских калибровочных коэффициентов из ППЗУ модуля и запись их в соответствующие поля подструктуры **DAC_CHANNEL_CALIBRATION**. Данная функция встроена в функцию **LTR34_Open()**, т.е. при открытии модуля калибровочные коэффициенты считываются из ППЗУ автоматически. Однако, если по каким-то причинам это требуется, пользователь может в любой момент самостоятельно выполнить чтение коэффициентов путем вызова рассматриваемой функции.
Возвращаемое значение: код ошибки, тип INT. Если “0” – функция выполнилась без ошибок

Формат: LPCSTR LTR34_GetErrorString(INT ErrorCode)
Параметры: - **module** – указатель на управляющую структуру модуля, тип TLTR34;
Описание: Возвращает строку, описывающую ошибку, соответствующую коду **ErrorCode**
Возвращаемое значение: строка, соответствующая данному коду ошибки

4 Конфигурирование модуля

Перед запуском генерации данных и загрузкой их в FIFO-буфер ЦАПа необходимо выполнить конфигурацию модуля. Конфигурация заключается в присвоении нужных значений соответствующим полям управляющей структуры с последующей передачей их модулю путем вызова функции **LTR34_Config()**.

4.1 Применение аппаратного сброса модуля

Для корректной генерации данных (особенно в потоковом режиме) перед каждым запуском генерации данных *обязательно* следует выполнять аппаратный сброс модуля (функция **LTR34_Reset()**), конфигурацию (**LTR34_Config()**), подготовку и загрузку данных (**LTR34_ProcessData()**, **LTR34_Send()**) и старт генерации **LTR34_DACStart()**.

4.2 Установка количества активных каналов модуля

В модуле LTR34 возможно использование одновременно 1-го, 2-х, 4-х или 8-ми каналов. Т.е. при работе с модулем не может быть, например, 3 или 5 активных каналов. Возможное количество используемых каналов – только из ряда 1, 2, 4 и 8, и оно определяется значением поля **ChannelQnt** управляющей структуры модуля. Например, чтобы использовать четыре канала, необходимо выполнить присваивание: **module.ChannelQnt**=4. Если значение количества активных каналов задано неверно, функция **LTR34_Config()** завершится с соответствующей ошибкой.

4.3 Установка частоты дискретизации ЦАП

Частота дискретизации ЦАП модуля LTR34 определяется значениями поля **FrequencyDivisor** управляющей структуры модуля и числом активных каналов.

Общая частота дискретизации (частота вывода очередного сэмпла) равна

Fs=2 МГц/(64 - FrequencyDivisor).

Чтобы определить частоту дискретизации для каждого канала, необходимо значение общей частоты дискретизации поделить на количество активных каналов:

Fch=Fs/ NumberOfChannel=2 МГц / ((64 - FrequencyDivisor)* NumberOfChannel).

Значение поля ***FrequencyDivisor*** *не может превышать 60.* Т.е. максимальная общая частота дискретизации модуля – 500 кГц. При работе с одним активным каналом максимальная частота дискретизации для этого канала равна общей и составляет 500 кГц. Если задействовано 8 каналов, то максимальная частота дискретизации одного канала составляет

2 МГц / (64-60)*8= 62500 Гц.

Минимальная общая частота дискретизации составляет, очевидно, 31250 Гц. При использовании всех 8-ми каналов минимальная частота дискретизации каждого канала составляет 3906.5 Гц.

После работы функции **LTR34_Config()** поле **FrequencyDAC** управляющей структуры модуля примет значение *общей частоты дискретизации*.

4.4 Установка режима генерации

Модуль LTR34 имеет два режима генерации сигнала: автогенерация (кольцевой) и потоковый. Более подробно о генерации данных см в гл. 5.

- Режим *автогенерации* устанавливается путем присваивания значения **true** полю **RingMode** управляющей структуры модуля.
- *Потоковый* режим устанавливается путем присваивания значения **false** полю **RingMode** управляющей структуры модуля.

4.5 Конфигурация старта генерации данных

Запуск генерации данных модулем LTR43 можно осуществить двумя способами: внутренний старт (по команде пользовательского программного обеспечения) и внешний старт (по внешнему аппаратному синхроимпульсу). Тип старта устанавливается при помощи поля **ExternalStart** управляющей структуры модуля:

внутренний старт устанавливается присваиванием полю **ExternalStart** значения **false**.
внешний старт устанавливается присваиванием полю **ExternalStart** значения **true**.

4.6 Установка типа подтверждения

В процессе генерации данных модуль периодически высылает в крейт-контроллер и ПК ответные слова-подтверждения, содержащие информацию о накопленных данных, признаки переполнения или опустошения FIFO-буфера. Программно можно установить два типа подтверждения:

- **Периодическая отправка информации о состоянии FIFO-буфера.** Модуль периодически высылает информацию о состоянии FIFO-буфера. Эта информация принимается *только сервером* и не доходит до пользовательского ПО. На ее основе сервер отслеживает состояние FIFO-буфера и разрешает или запрещает выполнение загрузки следующей партии данных в буфер. Если текущая степень заполнения буфера не позволяет загрузить данные, сервер ставит запрос на отправку в очередь. Указанный тип подтверждения устанавливается путем присвоения значения **false** полю **AcknowledgeType** управляющий структуры модуля. *Этот режим является предпочтительным для применения в пользовательском программном обеспечении*.

- **На каждое слово.** В процессе генерации данных после каждого выведенного сэмпла модуль высылает подтверждение. Это происходит независимо от того, какой режим генерации установлен: автогенерация или потоковый. Каждое слово-подтверждение является «эхом» выведенного сэмпла. Данный тип подтверждения устанавливается путем присвоения значения **true** полю **AcknowledgeType** управляющий структуры модуля.

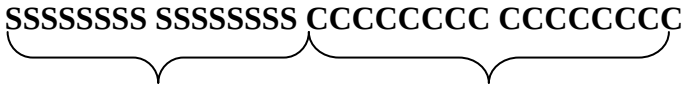
*После определения всех вышеперечисленных полей следует обязательно вызвать функцию **LTR34_Config()**, чтобы передать все параметры модулю. Если этого не сделать, то генерация данных модулем может быть некорректной.*

Выполнив конфигурирование модуля, можно приступать к запуску генерации сигнала.

4.7 Создание таблицы логических каналов

Одно из полей структуры описания модуля – Таблица Логических Каналов **LChTbl[]**– представляет собой массив, содержащий информацию об используемых физических каналах модуля и соответствующих им выходах (масштабирующем или нет). В отличие от физического канала, **Логический канал** представляет собой 32-битное слово специального формата, содержащее информацию о соответствующем физическом канале. **Таблицей Логических Каналов** называется массив, каждый элемент которого представляет собой логический канал. Совокупность элементов этой таблицы описывает те физические каналы, для которых ЦАП будет генерировать сигнал. Для физических каналов, не представленных в Таблице Логических Каналов, генерация сигнала производиться не будет. Заполнение этого массива (Таблицы Логических Каналов) *осуществляется пользователем*. Для создания логического канала следует вызвать функцию **LTR34_CreateLChannel(),** где в параметрах нужно указать номер физического канала, связываемого с данным логическим, а также используемый выход для этого физического канала (прямой или масштабирующий 1:10). Функция возвращает сформированное значение логического канала, которое следует записать в Таблицу *вручную*. В Таблице Логических Каналов элементы, соответствующие различным физическим каналам модуля, **не обязательно** должны располагаться последовательно, в порядке возрастания номеров физических каналов. С любым логическим каналом может быть связан любой физический, от 1-го до 8-го (4-го). Однако *повторы недопустимы*. Т.е. два логических канала *не могут быть связаны с одним и тем же физическим*. Количество инициализированных элементов Таблицы Логических Каналов должно соответствовать значению поля **ChannelQnt** управляющей структуры модуля. Важно помнить, что Таблица Логических Каналов **не должна содержать неинициализированных полей в диапазоне индексов от 0 до LChQnt-1.** Т.е. если заявлено количество логических каналов, равное значению поля **LChQnt**, то все элементы массива **LchTable[]** с индексами от 0 до **LChQnt**-1 должны быть заполнены верными значениями логических каналов. В противном случае данные, выдаваемые модулем, будут некорректными. **Всегда следует определять количество задействованных логических каналов и следить за тем, чтобы все элементы Таблицы Логических Каналов и индексами от 0 до LChQnt-1 были инициализированы верными значениями!**

Ниже приводится формат логического канала (32-битное слово):



Биты 16-31 Биты 0..15

Биты C <0..15> - номер физического канала, связываемого с данным логическим (нумерация с единицы);

Биты C <16..31> - Тип выхода для данного физического канала (прямой или масштабирующий).

Формат логического канала здесь приведен как справочная информация. Пользователю не требуется самостоятельно формировать значения этих слов, это делается автоматически при помощи функции **LTR34_CreateLChannel().** Пользователю только следует вызвать эту функцию с требуемыми параметрами, а возвращаемое значение и будет являться сформированным логическим каналом, которое затем следует записать в

Таблицу Логических Каналов под требуемым индексом. *Следует иметь ввиду, что физические каналы при вызове данной функции нумеруются с единицы (1-8), а индексы элементов Таблицы Логических каналов – с нуля (0-7).*

Например, в 8-канальном ЦАП LTR34-8 будем использовать 4 физических канала: 1,3,6 и 7. Причем в каналах 1 и 7 будем использовать прямые выходы, а в 3 и 6 – масштабирующие (1:10). Тогда последовательность создания Таблицы Логических Каналов будет выглядеть следующим образом:

TLTR34 conf;

conf.ChannelQnt=4; // Количество активных каналов
conf.LChTbl[0]=LTR34_CreateLChannel(1,0); // физ. канал 1, прямой выход
conf.LChTbl[0]=LTR34_CreateLChannel(3,1); // физ. канал 3, выход 1:10
conf.LChTbl[0]=LTR34_CreateLChannel(6,1); // физ. канал 6, выход 1:10
conf.LChTbl[0]=LTR34_CreateLChannel(7,0); // физ. канал 7, прямой выход

5 Особенности генерации сигнала модулем LTR34

5.1 Формирование массива данных для загрузки в FIFO-буфер модуля

Сигнал, генерируемый модулем, определяется массивом сэмплов, который должен быть обязательно загружен в FIFO-буфер модуля перед стартом генерации. Массив формируется на верхнем уровне и затем загружается в буфер при помощи функции **LTR34_Send()**.

Однако значения 16-битных кодов или напряжения перед отправкой в модуль следует предварительно перевести в формат слов-данных в соответствии с протоколом обмена в системе LTR. Для этого предназначена функция
LTR34_ProcessData(TLTR34 *module, double *src, DWORD *dest, DWORD size, bool volt).

Параметры функции:

- **module** – экземпляр управляющей структуры модуля.
- **data** – указатель на массив, который содержит данные, определяющие уровень сигнала, генерируемого модулем. Данные могут быть заданы двумя способами:

  Непосредственно в 16-битных кодах (от –32768 (FFFF hex) до 32767 (7FFF hex)).

  В вольтах. При этом они представляют собой непосредственно значения генерируемого напряжения. *Следует учесть, что если для данного физического канала выбран масштабирующий выход 1:10, то напряжение в вольтах должно задаваться именно таким, каким оно должно быть сгенерировано на этом выходе.* Если, например, требуется вывести на масштабирующий выход 0,5 В, то следует задавать именно 0,5 В. Хотя при этом на прямой выход того же канала будет выведено 5 В. Также в этом случае буду применяться калибровочные коэффициенты, соответствующие именно выходу 1:10.
- **size** – размер массива, отправляемого в FIFO-буфер ЦАП.
- **volt** – определяет, в каких единицах заданы исходные данные. Если **volt**=0, то данные в массиве **src** представляют собой непосредственно коды для вывода на ЦАП. Если **volt**=1, то исходные данные заданы в вольтах и представляют собой уровень генерируемого напряжения.

Данные в исходном массиве **src** должны быть расположены *по кадрам*. **Кадр** представляет собой последовательность слов данных, предназначенных для одного цикла генерации для всех задействованных каналов. Т.е. каждый кадр будет содержать слова-данные для всех определенных логических каналов. В каждом кадре данные должны располагаться последовательно в порядке следования логически каналов в Таблице Логических Каналов. Каждое слово в массиве **src** соответствует определенному логическому каналу, определяемому смещением элемента от начала кадра. Количество кадров в массиве **src** будет равно количеству элементов этого массива, деленному на количество активных каналов, т.е. на значение поля **ChannelQnt**. Например, если определено 4 логических канала, а размер массива 1000 элементов, то количество кадров в массиве будет равно 1000/4=250. Как уже говорилось выше, в зависимости от значения параметра **volt,** значения массива **src** могут представлять собой коды или значения напряжения в вольтах. Но в любом случае этот массив имеет тип **double.**

Диапазон значений данных от –32768 до 32767. В процессе генерации сигнала (с учетом калибровочных коэффициентов) коду –32768 соответствует значение напряжения –10 В, коду 32767 – напряжение +10 В. Отсюда можно вывести, что значение генерируемого напряжения для произвольного кода code будет равно **V=code*20 / 65535.** А код, соответствующий требуемому напряжению V будет рассчитываться следующим образом: **code=V*65535 / 20** В. В случае использования масштабирующего выхода **V=code*2.0/65535 В** и **code=V*65535/2.0**.

Приведем пример. Если определены следующие логические каналы (как в примере выше):

TLTR34 conf;

conf.ChannelQnt=4; // Количество активных каналов

conf.LChTbl[0]=LTR34_CreateLChannel(1,0); // физ. канал 1, прямой выход
conf.LChTbl[0]=LTR34_CreateLChannel(3,1); // физ. канал 3, выход 1:10
conf.LChTbl[0]=LTR34_CreateLChannel(6,1); // физ. канал 6, выход 1:10
conf.LChTbl[0]=LTR34_CreateLChannel(7,0); // физ. канал 7, прямой выход,

а массив **src** содержит следующие элементы:

0x0;
0x32767;
0x20000;
0x5439;
0x3445;
0xFFAC;
0x46C1;
0x9014,

то в данном массиве, очевидно, два кадра. Каждый кадр содержит 4 элемента, т.к. количество активных каналов в этом примере 4.

Значение 0x0 соответствует логическому каналу 0, будет выведено на физ. канал 1,
Значение 0x32767 соответствует логическому каналу 1, будет выведено на физ. канал 3,
Значение 0x20000 соответствует логическому каналу 2, будет выведено на физ. канал 6,
Значение 0x5439 соответствует логическому каналу 3, будет выведено на физ. канал 7,

Значение 0x3445 соответствует логическому каналу 0, будет выведено на физ. канал 1,
Значение 0xFFAC соответствует логическому каналу 1, будет выведено на физ. канал 3,
Значение 0x46C1 соответствует логическому каналу 2, будет выведено на физ. канал 6,
Значение 0x9014 соответствует логическому каналу 3, будет выведено на физ. канал 7.

Функция **LTR34_ProcessData()** сформирует выходной массив **dest**, готовый для отправки в FIFO-буфер модуля. В этом массиве слова-данные аналогично будут располагаться по кадрам, но в каждое слово будут добавлены служебные поля, и все элементы массива **dest** будут представлять собой готовые данные для отправки в FIFO-буфер модуля. Если установлен режим автогенерации, массив **dest** следует загрузить один раз в FIFO-буфер модуля, а затем выполнить старт генерации. Если выбран потоковый режим, то массив **dest** следует загрузить перед стартом сбора данных, а затем периодически формировать новые данные и подгружать в FIFO-буфер по мере вывода на ЦАП.

5.2 Расширение возможностей формирования массива данных

Использование Таблицы логических каналов и применение функции **LTR34_ProcessData()** – удобный способ подготовки слов-данных для отправки в модуль. При этом пользователю не требуется вникать в формат этих слов, а оперировать только значениями кода сигнала.

Однако в этом случае присутствует следующее ограничение: Таблица Логических Каналов создается один раз перед конфигурацией и стартом сбора данных. Таким образом, процесс генерации будет осуществляться для одних и тех же каналов и в одной и той же последовательности, определяемой Таблицей. Однако модуль позволяет на каждый цикл генерации выводить данные в иные каналы, чем были выведены в предыдущем цикле. Чтобы использовать эту возможность, пользователю требуется самостоятельно формировать массив для отправки данных в модуль согласно требуемому формату и вручную определять канал для вывода каждого сэмпла с последующей отправкой массива в модуль функцией LTR34_Send. Также пользователю самому необходимо применять калибровочные коэффициенты, считывая их из массива **DAC_Calibration**. Но в этом случае выбор каналов ничем не ограничен. Формат слов-данных модуля LTR34 приводится нидже.

Слова-данные, загружаемые в FIFO модуля, имеют следующий формат:

DDDDDDDD DDDDDDDD SSSSSSSS 0000CCC0

- Биты D – 16-битный код выводимого сигнала (от –32768 до 32767).
- Биты S – служебная информация (номер посадочного места и т.д.)
- Биты C – номер физического канала (от 0 до 7, соответствует каналам 1-8)

Вручную устанавливая в битах **D** требуемый код, а в битах **C** номер физического канала (нумерация здесь с 0!), можно формировать массив данных для вывода в произвольные каналы. Этот массив можно непосредственно засылать в буфер модуля функцией **LTR34_Send(),** минуя создание Таблицы Логических Каналов и вызов функции **LTR34_ProcessData()**. Однако количество задействованных каналов обязательно должно быть определено в поле **ChannelQnt** управляющей структуры модуля.

Калибровочные коэффициенты при таком способе формирования данных пользователю также придется применять самостоятельно. Готовый откорректированный код равен:

k*code+b. Здесь **code** – значение кода, **k**-заводской коэффициент усиления, **b** – заводской коэффициент смещения.

Заводские калибровочные коэффициенты считываются из ППЗУ модуля в массив **DacCalibration** управляющей структуры при открытии модуля функцией **LTR34_Open()**. Расположение коэффициентов в массиве следующее:

- **DacCalibration.FactoryCalibrScale[2*i]** – коэффициенты усиления для выходов 1:1. i – номер физического канала (нумерация с 0).
- **DacCalibration.FactoryCalibrScale[2*i+1]** – коэффициенты смещения для выходов 1:10 1:10. i – номер физического канала (нумерация с 0).
- **DacCalibration.FactoryCalibrOffset[2*i]** – коэффициенты усиления для выходов 1:1. i – номер физического канала (нумерация с 0).
- **DacCalibration.FactoryCalibrOffset[2*i+1]** – коэффициенты смещения для выходов 1:10 1:10. i – номер физического канала (нумерация с 0).

5.3 Загрузка данных в FIFO-буфер модуля

Если выбран режим *автогенерации* (RingMode=1), то перед стартом генерации данные следует ***один раз*** загрузить в FIFO-буфер модуля. После загрузки специальный счетчик автоматически определяет, сколько данных было загружено. Это значение равно, очевидно размеру объема FIFO-буфера, занятого данными. При автогенерации после вывода последнего сэмпла на ЦАП модуль автоматически переходит к выводу первого сэмпла.

При *потоковом* режиме массив с данными следует загрузить не только перед стартом генерации, но и периодически подгружать в FIFO новые массивы данных по мере вывода на ЦАП.

Загрузка данных в FIFO-буфер модуля выполняется при помощи функции **LTR34_Send(TLTR34 *module, DWORD *data, DWORD size, DWORD timeout).**

Параметр **module** - указатель на управляющую структуру модуля;

Параметр **data** - указатель на массив с данными, подлежащими выводу на ЦАП. Этот массив должен соответствовать выходному массиву **dest** функции **LTR34_ProcessData()**. Пользователь также может сформировать этот массив вручную с учетом формата слов-данных, выводимых на ЦАП.

Параметр **size** - размер массива data;

Параметр **timeout** представляет собой максимальное время в миллисекундах, в течение которого функция **LTR34_SendData()** может выполняться. Если массив данных будет передан за время, меньшее, чем **timeout**, то функция завершится тут же после передачи, не дожидаясь истечения таймаута. Если время **timeout** истекло, то выполнение функции завершается независимо от того, сколько данных было передано.

Возвращаемое значение функции **LTR34_Send()** – количество слов команд или данных, отправленных модулю. Если функция вернула отрицательное значение, это свидетельствует об ошибке, а возвращаемое значение представляет собой код ошибки.

5.4 Применение калибровочных коэффициентов

Поле **UseClb** управляющей структуры модуля определяет, следует ли задействовать фабричные калибровочные коэффициенты при формировании и генерации данных. Если значение поля равно **true**, то к каждому сэмплу будут применены фабричные калибровочные коэффициенты, считанные из подструктуры **DacCalibration** управляющей структуры модуля (напомним, что в указанную подструктуру калибровочные коэффициенты загружаются из ППЗУ модуля во время работы функции **LTR34_Open()**). Если значение поля равно **false**, то калибровочные коэффициенты применяться не будут. Смысл генерации сигнала без применения калибровочных коэффициентов весьма

сомнителен, т.к. погрешность в этом случае достигает значительных величин (несколько процентов), поэтому рекомендуется всегда применять заводские калибровочные коэффициенты.

Калибровочные коэффициенты применяются с учетом того, какой выход для данного физического канала определен в соответствующем логическом. Если прямой – применяются калибровочные коэффициенты, рассчитанные для прямого выхода. При этом на выходе 1:10 значение сигнала будет неточным. Если масштабирующий - применяются калибровочные коэффициенты, рассчитанные для выхода 1:10. При этом на прямом выходе значение сигнала будет неточным.

6 Генерация данных и получение ответных слов-подтверждений

6.1 Запуск и остановка генерации сигнала

Как указывалось выше, для модуля LTR34 существует два способа запуска генерации: внутренний старт и внешний старт.

Если выбран **внутренний старт** (поле управляющей структуры **ExternalStart**=0), то функция **LTR LTR34_DACStart()** ***немедленно*** запускает генерацию. Напомним, что перед вызовом этой функции ***обязательно*** следует выполнить конфигурацию модуля при помощи функции **LTR34_Config()** и загрузку FIFO-буфера данными при помощи функции **LTR34_SendData()**.

Если выбран **внешний старт** (поле управляющей структуры **ExternalStart**=1), то также должна быть вызвана функция **LTR34_DACStart()**, однако непосредственно сам старт генерации начнется только после прихода фронта сигнала START, подаваемого на соответствующие клеммы внешнего разъема модуля (подробнее об этом см. в документе Крейтовая система LTR. Руководство пользователя). Таким образом, при использовании внешнего старта функция **LTR34_DACStart()** запускает не генерацию данных, а ожидание внешнего импульса START, после прихода которого начнется процесс генерации.

После старта генерации модуль переходит в режим РАБОТА, флаг **DACRunning** управляющей структуры модуля принимает значение **true**, и модуль с этого момента не будет реагировать ни на какие команды, кроме остановки генерации данных.

Генерация сигнала останавливается при помощи вызова функции **LTR34_DACStop()**.

6.2 Подкачка данных и получение подтверждений в процессе генерации сигнала

При использовании режима автогенерации периодического сигнала данные должны быть загружены в FIFO-буфер модуля ***один раз*** – перед стартом генерации. Это, как указывалось выше, осуществляется функцией **LTR34_Send()**. В процессе работы ЦАП дополнительно подгружать данные в FIFO-буфер не требуется.

В потоковом режиме необходимо периодичски подгружать данные в буфер по мере его опустошения, поэтому в процессе работы ЦАП требуется постоянно отслеживать состояние FIFO-буфера и, если требуется, загрузить туда новую порцию данных. Для отслеживания состояния FIFO-буфера в модуле LTR34 предусмотрены два типа подтверждений: **периодическая посылка статуса FIFO-буфера** и **подтверждение на каждое выведенное слово** (режим эха).

Если выбран режим периодической посылки статуса FIFO-буфера, то модуль периодически высылает в крейт информацию о заполненном объеме FIFO-буфера (в

килобайтах). Эта информация анализируется ***только*** LTR-сервером и до пользователя не доходит. LTR-сервер на основе этой информации принимает решение, когда следует загрузить в FIFO следующую порцию данных, чтобы не допустить опустошение или переполнение FIFO-буфера. Если имеется запрос на загрузку данных (вызвана функция **LTR43_Send()**), то данные фактически будут отправлены только тогда, когда информация о заполненности FIFO позволит серверу это сделать. И именно после отправки данных функция **LTR_Send()** завершится. При потоковом выводе с применением периодической посылки статуса целесообразно функцию **LTR34_Send()** использовать в цикле. Таким образом, следующая итерация начнется не раньше момента, когда очередная порция данных будет оправлена в буфер. Формат подтверждения в этом случае следующий:

0000ZZZZ ZZZZZZZZ SSSSSSSS 110ExxxF

- биты **Z** – значение, равное количеству накопленных сэмплов в FIFO-буфере, округленное до килосэмплов. Максимальное значение – 2048. Значение 0 соответствует количеству менее одного килосэмпла;
- бит **F** – признак переполнения FIFO-буфера. Если бит равен 1 - буфер переполнен, 0 – переполнения буфера не произошло;
- бит **E** – признак опустошения буфера. Если бит равен 1 – буфер пуст. 0 – опустошения буфера не произошло.

Еще раз напомним, что эти слова-подтверждения анализируются только сервером и не доходят до пользовательского ПО. Их формат приводим как справочную информацию, которая может быть полезна в экзотических случаях работы без LTR-сервера.

Внимание! Режим работы с периодической посылкой статуса FIFO-буфера является предпочтительным!

Если выбран режим **подтверждения на каждое слово** (поле управляющей структуры **AcknowledgeType**=0), то слова-подтверждения приходят после вывода ***каждого сэмпла***. Другими словами, каждое слово-подтверждение представляет собой эхо соответствующего сэмпла, выведенного на ЦАП. Кроме того, каждое подтверждение содержит информацию о переполнении или, наоборот, опустошении FIFO-буфера. Формат слова-подтверждения более подробно:

DDDDDDDD DDDDDDDD SSSSSSSS 111ECCCF

- биты **D** – код соответствующего сэмпла, выведенный на ЦАП;
- биты **S** – служебная информация (бит-признак команды, номер посадочного места);
- бит **F** – признак переполнения FIFO-буфера. Если бит равен 1 - буфер переполнен, 0 – переполнения буфера не произошло;
- биты **CCC** – номер канала ЦАП соответствующего сэмпла;
- бит **E** – признак опустошения буфера. Если бит ране 1 – буфер пуст. 0 – опустошения буфера не произошло.

Принимая поток подтверждений, можно отслеживать, какие сэмплы уже выведены на ЦАП и на основе этой информации принимать решение о подкачке новой порции данных в FIFO-буфер ЦАПа. Также полезно отслеживать возможные состояния переполнения и опустошения буфера по значениям битов F и E.

При режиме с подтверждениями на каждое слово потоковую генерацию данных целесообразно осуществлять при помощи двух программных потоков (thread). Один поток будет предназначен для периодической отправки данных в модуль при помощи

функции **LTR34_SendData()**, а второй – для получения подтверждений при помощи функции **LTR34_Receive()**, отслеживания состояния буфера и на основе этой информации принимать решение об отправке очередной порции данных в модуль.

Внимание! Режим работы с подтверждением на каждое слово является тестовым, и использовать его в пользовательском ПО не рекомендуется!

7 Приложения

7.1 Приложение 1. Пример программы

Программа представляет собой консольное приложение, написанное в среде Microsoft Visual C++ 2005.

```
#include "stdafx.h";
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include "ltr\\include\\ltr34api.h";

void main()
{
TLTR34 conf;
int j;
DWORD size;
int SizeSent;
int err=0;
INT SlotNum; // Номер посадочного места
CHAR FPGA_Version[255]; // В эту переменную запишем версию прошивки ПЛИС
CHAR SerialNum[255]; // В эту переменную запишем серийный номер модуля
CHAR ModuleName[255]; // В эту переменную запишем имя модуля
CHAR ErrorString[255]; // Строка для вывода описания ошибки
CHAR MsgString[255]; // Строка для вывода сообщений на консоль

/* Инициализируем канал связи с модулем. Поля структуры описания модуля
заполняются значениями по умолчанию. */

err=LTR34_Init(&conf);
if(err)
  {
      strcpy(ErrorString, (char *) LTR34_GetErrorString(err));
      CharToOem(ErrorString,ErrorString);
      printf("%s",ErrorString);
      LTR34_Close(&conf);
      Sleep(3000);
      return;
   }

// Открываем канал связи с модулем. Серийный номер крейта и сетевой адрес -
по умолчанию

// Номер слота - 1;
SlotNum=0;
// Полный путь файла с прошивкой ПЛИС - C://ltr51.ttf
err=LTR34_Open(&conf, SADDR_DEFAULT, SPORT_DEFAULT,"", SlotNum);
if(err)
  {
      if(err==LTR_WARNING_MODULE_IN_USE)
       {
```

```
            strcpy(ErrorString, (char *) LTR34_GetErrorString(err));
           CharToOem(ErrorString,ErrorString);
           printf("%s",ErrorString);
         }
        else
        {
         strcpy(ErrorString, (char *) LTR34_GetErrorString(err));
         CharToOem(ErrorString,ErrorString);
         printf("%s",ErrorString);
         return;
        }
    }
/* Заполнение полей для информации. Здесь сделано только для демонстрации */

strcpy(FPGA_Version, conf.ModuleInfo. FPGA_Version);
strcpy(MsgString,"Версия прошивки ПЛИС: ");
strcat(MsgString, FPGA_Version);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

strcpy(SerialNum, conf.ModuleInfo.Serial);
strcpy(MsgString,"Серийный номер модуля: ");
strcat(MsgString, SerialNum);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

strcpy(ModuleName, conf.ModuleInfo.Name);
strcpy(MsgString,"Имя модуля: ");
strcat(MsgString, ModuleName);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

// Заполняем Таблицу Логических каналов.
// Во всех каналах используем немасштабирующий выход 1:1

conf.ChannelQnt=2; // Задействуем 2 канал. Пусть это будут 1-й и 3-й физ.
каналы
conf.LChTbl[0]=LTR34_CreateLChannel(1,0);
conf.LChTbl[1]=LTR34_CreateLChannel(3,0);


// Заполняем поля управляющей структуры модуля соответственно режиму

conf.FrequencyDivisor=40; // делитель частоты дискретизации

conf.UseClb=true; // Используем заводские калибровочные коэффициенты
conf.AcknowledgeType=false; // тип подтверждения - периодические
conf.ExternalStart=false;   // внешний старт отключен
conf.RingMode=true;         // включен режим автогенерации

// Перед конфигурацией ОБЯЗАТЕЛЬНО выполняем сброс модуля

err=LTR34_Reset(&conf);
if(err)
  {
      strcpy(ErrorString, (char *) LTR34_GetErrorString(err));
      CharToOem(ErrorString,ErrorString);
      printf("%s",ErrorString);
      LTR34_Close(&conf);
      Sleep(3000);
      return;
    }
// Выполняем конфигурацию модуля
err=LTR34_Config(&conf);
```

```
if(err)
  {
      strcpy(ErrorString, (char *) LTR34_GetErrorString(err));
      CharToOem(ErrorString,ErrorString);
      printf("%s",ErrorString);
      LTR34_Close(&conf);
      Sleep(3000);
      return;
   }

sprintf(MsgString,"Частота генерации: %f Гц", conf.FrequencyDAC);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

   double *Code= new double[10000];
   DWORD *ArrayToSend=new DWORD[10000];

   j=0;
   size=0;
            // заполняем значение массива для отсылки - чтобы генерилась пила
    for (double i=0.0;i<8.0;i+=0.01)
       {
      // Формируем массив. Сигнал "Пила" от 0 до 8 В. Шаг 0.01 В
        Code[j]= i; // Для первого лог. канала. Физический канал 1
        j++;

        Code[j]= i; // Для второго лог. канала. Физический канал 3
        j++;
       }
size=j; // Размер массива
// Вызываем функцию подготовки слов-данных. ArrayToSend - массив, готовый к
отправке.
// Указываем, что сигнал задан В ВОЛЬТАХ!
err=LTR34_ProcessData(&conf, Code, ArrayToSend, size, true);
if(err)
  {
      strcpy(ErrorString, (char *) LTR34_GetErrorString(err));
      CharToOem(ErrorString,ErrorString);
      printf("%s",ErrorString);
      Sleep(3000);
      goto exit;
   }
// Отправляем данные в модуль
SizeSent=LTR34_Send(&conf, ArrayToSend, size, 5000);
if(SizeSent!=(int) size)
{
 if(SizeSent < 0)
 {
  sprintf(MsgString,"%s", "Ошибка при отправке данных модулю!");
  CharToOem(MsgString,MsgString);
  printf("%s\n",MsgString);
  }
 else
 {
  sprintf(MsgString, "Ошибка! Количество отправленных сэмплов не равно
заданному!");
  CharToOem(MsgString,MsgString);
  printf("%s\n",MsgString);
  }
  goto exit;
 }

// Запускаем генерацию
```

```
err=LTR34_DACStart(&conf);
if(err)
  {
      strcpy(ErrorString, (char *) LTR34_GetErrorString(err));
      CharToOem(ErrorString,ErrorString);
      printf("%s",ErrorString);
      Sleep(3000);
      goto exit;
   }
// На физических каналах 1 и 3 появляется одинаковая "Пила"

 sprintf(MsgString,"%s", "Идет генерация данных в режиме автогенерации в
течение 5 с");
 CharToOem(MsgString,MsgString);
 printf("%s\n",MsgString);

 Sleep(5000);

 // Останавливаем генерацию

 err=LTR34_DACStop(&conf);
if(err)
  {
      strcpy(ErrorString, (char *) LTR34_GetErrorString(err));
      CharToOem(ErrorString,ErrorString);
      printf("%s",ErrorString);
      Sleep(3000);
      goto exit;
   }

sprintf(MsgString,"%s", "Генерация остановлена");
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

sprintf(MsgString,"Сбор данных окончен. Нажмите любую клавишу для выхода.");
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);
getchar();

exit:

if(ArrayToSend!=NULL)
 delete(ArrayToSend);

if(Code!=NULL)
 delete(Code);

LTR34_Close(&conf);

return;

}
```